Applicative Bisimilarities for Call-by-Name and Call-by-Value $\lambda\mu$-Calculus

Dariusz Biernacki [1]

Institute of Computer Science
University of Wrocław
Wrocław, Poland

Sergueï Lenglet [2]

LORIA
Université de Lorraine
Nancy, France

Abstract

We propose the first sound and complete bisimilarities for the call-by-name and call-by-value untyped $\lambda\mu$-calculus, defined in the applicative style. We give equivalence examples to illustrate how our relations can be used; in particular, we prove David and Py's counter-example, which cannot be proved with Lassen's preexisting normal form bisimilarities for the $\lambda\mu$-calculus.

Keywords: contextual equivalence, applicative bisimulation, continuation, control operator

1 Introduction

Contextual equivalence [13] is considered as the most natural behavioral equivalence in languages based on the λ-calculus. Two terms are equivalent if an outside observer cannot tell them apart when they are evaluated within any *context* (a term with a hole). However, the quantification over contexts makes proving the equivalence of two given programs cumbersome. Consequently, other characterizations of contextual equivalence are sought for, such as coinductively defined *bisimilarities*.

Several kinds of bisimilarity have been proposed, such as, e.g., *applicative* bisimilarity [1], which relates terms by reducing them to values (if possible), and then compares these values by applying them to an arbitrary argument. The idea is the same for *environmental* bisimilarity [18], except the values are tested with arguments built from an environment, which represents the knowledge of an observer

[1] Email: dabi@cs.uni.wroc.pl
[2] Email: serguei.lenglet@univ-lorraine.fr

about the tested terms. Finally, *normal form* bisimilarity [11] (initially called *open* bisimilarity [17]) reduces open terms to normal forms and then compares their subterms. Applicative and environmental bisimilarities still contain some quantification over arguments, and usually coincide with contextual equivalence. In contrast, normal form bisimilarity is easier to use, as its definition does not contain any quantification over arguments, but it is generally not *complete*, i.e., there exist equivalent terms that are not normal form bisimilar.

This article treats the behavioral theory of the untyped $\lambda\mu$-calculus [15]. The $\lambda\mu$-calculus provides a computational interpretation of classical natural deduction and thus extends the Curry-Howard correspondence from intuitionistic to classical logic. Operationally, the reduction rules of the calculus express not only function applications but also capturing of the current context of evaluation. Therefore, when considered in the untyped setting, the calculus offers an approach to the semantics of abortive control operators such as *call/cc* known from the Scheme programming language and it may be viewed as a closely related alternative to Felleisen and Hieb's syntactic theory of control [6].

So far no characterization of contextual equivalence has been proposed for either call-by-value or call-by-name $\lambda\mu$-calculus. Lassen defined normal form bisimilarities for call-by-name weak-head reduction [10], for head reduction [12], and, with Støvring, for call-by-value weak-head reduction [19] that are not complete. However, normal form bisimilarity is complete for the $\Lambda\mu$-calculus [5] with head reduction [12], and also for the $\lambda\mu$-calculus with store [19] under call-by-value weak-head reduction. Lassen also defined an incomplete applicative bisimilarity for call-by-name weak-head reduction in [10]. A definition of applicative bisimilarity has also been proposed for a call-by-value typed μPCF [14], but the resulting relation is neither sound nor complete.

In this work, we propose the first characterizations of contextual equivalence for $\lambda\mu$-calculus for both call-by-name and call-by-value weak-head reduction semantics. The applicative bisimilarities we define are harder to use than Lassen's normal form bisimilarity to prove the equivalence of two given terms, but because they are complete, we can equate terms that cannot be related with normal form bisimilarity, such as David and Py's counter-example [4]. Even though the two applicative bisimilarities we define are built along the same principles, the relation we obtain in call-by-value is much more difficult to use than the one for call-by-name. However, we provide counter-examples showing that simplifying the call-by-value case so that it matches the call-by-name one leads to an unsound definition.

The paper is organized as follows. We first discuss the behavioral theory of the call-by-name (abbreviated as CBN) $\lambda\mu$-calculus in Section 2. We propose a notion of contextual equivalence (in Section 2.2) which observes top-level names, and we then characterize it with an applicative bisimilarity (Section 2.3). In particular, we compare our definition of bisimilarity with Lassen's work and we prove David and Py's counter-example using our relation. We then discuss call-by-value (CBV) in Section 3. We propose a definition of applicative bisimilarity (Section 3.2) which coincides with contextual equivalence. We also provide counter-examples showing that the definition cannot be naively simplified to match the one for call-by-name. Although the relation we obtain is harder to use than the one for call-by-name, we

can still prove some interesting equivalences of terms, as we demonstrate in Section 3.3. We conclude in Section 4. The accompanying research report [3] contains the proofs missing from this paper, and also discusses environmental bisimilarity for call-by-name.

2 Call-by-Name $\lambda\mu$-calculus

2.1 Syntax and Semantics

The $\lambda\mu$-calculus [15] extends the λ-calculus with named terms and a μ constructor that binds names in terms. We assume a set X of *variables*, ranged over by x, y, etc., and a distinct set A of *names*, ranged over by a, b, etc. Terms (T) and named terms (U) are defined by the following grammar:

$$\begin{aligned} \text{Terms:} \quad & t ::= x \mid \lambda x.t \mid t\,t \mid \mu a.u \\ \text{Named terms:} \quad & u ::= [a]t \end{aligned}$$

Values (V), ranged over by v, are terms of the form $\lambda x.t$. A λ-abstraction $\lambda x.t$ binds x in t and a μ-abstraction $\mu a.t$ binds a in t. We equate terms up to α-conversion of their bound variables and names, and we assume bound names to be pairwise distinct, as well as distinct from free names. We write $\mathsf{fv}(t)$ and $\mathsf{fv}(u)$ for the set of free variables of, respectively, t and u, and we write $\mathsf{fn}(t)$ and $\mathsf{fn}(u)$ for their set of free names. A term t or named term u is said *closed* if, respectively, $\mathsf{fv}(t) = \emptyset$ or $\mathsf{fv}(u) = \emptyset$. Note that a closed (named) term may contain free names. The sets of closed terms, closed values, and named terms are T^0, V^0, and U^0, respectively. In any discussion or proof, we say a variable or a name is *fresh* if it does not occur in any term under consideration.

We distinguish several kinds of contexts, represented outside-in, as follows:

$$\begin{aligned} \text{Contexts:} \quad & C ::= \Box \mid C\,t \mid t\,C \mid \lambda x.C \mid \mu a.\mathbb{C} \\ \text{Named contexts:} \quad & \mathbb{C} ::= [a]C \\ \text{CBN evaluation contexts:} \quad & E ::= \Box \mid E\,t \\ \text{Named evaluation contexts:} \quad & \mathbb{E} ::= [a]E \end{aligned}$$

The syntax of (named) evaluation contexts reflects the chosen reduction strategy, here call-by-name. Contexts can be filled only with a term t, to produce either regular terms $C[t]$, $E[t]$, or named terms $\mathbb{C}[t]$, $\mathbb{E}[t]$; the free names and free variables of t may be captured in the process.

We write $t_0\{t_1/x\}$ and $u_0\{t_1/x\}$ for the usual capture-avoiding substitution of terms for variables. We define the capture-avoiding substitution of named contexts for names, written $t\langle\mathbb{E}/a\rangle$ and $u\langle\mathbb{E}/a\rangle$, as follows. Note that the side-condition in the μ-binding case can always be fulfilled using α-conversion.

$$x\langle \mathbb{E}/a\rangle \stackrel{\text{def}}{=} x \qquad (\mu b.u)\langle \mathbb{E}/a\rangle \stackrel{\text{def}}{=} \mu b.u\langle \mathbb{E}/a\rangle \text{ if } b \notin \mathsf{fn}(\mathbb{E}) \cup \{a\}$$

$$(\lambda x.t)\langle \mathbb{E}/a\rangle \stackrel{\text{def}}{=} \lambda x.t\langle \mathbb{E}/a\rangle$$

$$(t_0\ t_1)\langle \mathbb{E}/a\rangle \stackrel{\text{def}}{=} t_0\langle \mathbb{E}/a\rangle\ t_1\langle \mathbb{E}/a\rangle \qquad ([b]t)\langle \mathbb{E}/a\rangle \stackrel{\text{def}}{=} \begin{cases} [b]t\langle \mathbb{E}/a\rangle & \text{if } a \neq b \\ \mathbb{E}[t\langle \mathbb{E}/a\rangle] & \text{if } a = b \end{cases}$$

We define the CBN reduction relation $\to_{\mathrm{n}}$ inductively by the following rules:

$$(\beta_n) \quad [a](\lambda x.t_0)\ t_1 \to_{\mathrm{n}} [a]t_0\{t_1/x\}$$

$$(\mu) \quad [a]\mu b.u \to_{\mathrm{n}} u\langle {}^{[a]}\Box/b\rangle$$

$$(app) \quad [a]t_0\ t_1 \to_{\mathrm{n}} u\langle {}^{[a]\Box\, t_1}/b\rangle \text{ if } [b]t_0 \to_{\mathrm{n}} u \text{ and } b \notin \mathsf{fn}([a]t_0\ t_1)$$

Reduction is defined on named terms only. The rule (β_n) is the usual call-by-name β-reduction. In rule (μ), the current continuation, represented by a, is captured and substituted for b in u. In an application (cf. rule (app)), we reduce the term t_0 in function position by introducing a fresh name b which represents the top level. We then replace b with $[a]\Box\, t_1$ in the result u of the reduction of $[b]t_0$. We can also express reduction with top-level evaluation contexts as follows.

Lemma 2.1 *$u \to_{\mathrm{n}} u'$ iff $u = \mathbb{E}[(\lambda x.t_0)\ t_1]$ and $u' = \mathbb{E}[t_0\{t_1/x\}]$, or $u = \mathbb{E}[\mu a.u'']$ and $u' = u''\langle \mathbb{E}/a\rangle$.*

Reduction is also compatible with evaluation contexts in the following sense.

Lemma 2.2 *If $u \to_{\mathrm{n}} u'$, then $u\langle \mathbb{E}/a\rangle \to_{\mathrm{n}} u'\langle \mathbb{E}/a\rangle$.*

We write $\to_{\mathrm{n}}^*$ for the transitive and reflexive closure of $\to_{\mathrm{n}}$, and we define the evaluation relation of the calculus as follows.

Definition 2.3 We write $u \Downarrow_{\mathrm{n}} u'$ if $u \to_{\mathrm{n}}^* u'$ and u' cannot reduce further.

If $u \Downarrow_{\mathrm{n}} u'$, then u' is a named value. If u admits an infinite reduction sequence, we say it *diverges*, written $u \Uparrow_{\mathrm{n}}$. For example, let $\Omega \stackrel{\text{def}}{=} (\lambda x.x\ x)\ (\lambda x.x\ x)$; then $[a]\Omega \Uparrow_{\mathrm{n}}$ for all a.

2.2 Contextual Equivalence

As in the λ-calculus, contextual equivalence in the $\lambda\mu$-calculus is defined in terms of convergence. However, unlike previous definitions [10,12], we define contextual equivalence on named terms first, before extending it to any terms.

Definition 2.4 Two closed terms u_0, u_1 are contextually equivalent, written $u_0 \approx_c u_1$, if for all closed contexts $\mathbb{C}$ and names a, there exist b, v_0, and v_1 such that $\mathbb{C}[\mu a.u_0] \Downarrow_{\mathrm{n}} [b]v_0$ iff $\mathbb{C}[\mu a.u_1] \Downarrow_{\mathrm{n}} [b]v_1$.

Note that we can plug only terms in a context, therefore we prefix u_0 and u_1 with a μ-abstraction. Definition 2.4 is not as generic as it could be, because we require the resulting named values to have the same top-level name b; a more general definition would simply say "$\mathbb{C}[\mu a.u_0] \Downarrow_{\mathrm{n}}$ iff $\mathbb{C}[\mu a.u_1] \Downarrow_{\mathrm{n}}$." Our definition is strictly finer than

the general one, because contexts cannot discriminate upon top-level names in some cases, as we can see with the next example.

Example 2.5 Let $\Theta \stackrel{\text{def}}{=} (\lambda x.\lambda y.y\ (x\ x\ y))\ (\lambda x.\lambda y.y\ (x\ x\ y))$ be Turing's CBN fixed-point combinator, and let $v \stackrel{\text{def}}{=} \lambda x.\lambda y.x$. The terms $u_0 \stackrel{\text{def}}{=} [a]\lambda x.\mu c.[b]\lambda y.\Theta\ v$ and $u_1 \stackrel{\text{def}}{=} [b]\lambda y.\Theta\ v$ are distinguished by Definition 2.4 if $a \neq b$, but we show they are related by the general contextual equivalence. To do so, we verify that $\mathbb{E}[\mu c.u_0] \Downarrow_{\mathrm{n}}$ iff $\mathbb{E}[\mu c.u_1] \Downarrow_{\mathrm{n}}$ holds for all $\mathbb{E}$ and c, and we can then conclude that u_0 and u_1 are in the general equivalence with David and Py's context lemma [4]. Let $\mathbb{E}$ be of the form $[d]E\ t$ for some d, E, t. Then $\mathbb{E}[\mu a.u_0] \Downarrow_{\mathrm{n}} [b]\lambda y.\Theta\ v$ and $\mathbb{E}[\mu a.u_1] \Downarrow_{\mathrm{n}} [b]\lambda y.\Theta\ v$, $\mathbb{E}[\mu b.u_0] \Downarrow_{\mathrm{n}} [a]\lambda x.\mu c.\mathbb{E}[\lambda y.\Theta\ v]$ and $\mathbb{E}[\mu b.u_1] \Downarrow_{\mathrm{n}} [d]\lambda y.\Theta\ v$, and finally $\mathbb{E}[\mu c.u_0] \Downarrow_{\mathrm{n}} u_0$ and $\mathbb{E}[\mu c.u_1] \Downarrow_{\mathrm{n}} u_1$ for $c \notin \{a, b\}$. The case $\mathbb{E} = [d]\Box$ is easy to check as well.

We choose Definition 2.4 because it gives more information on the behaviors of terms than the general equivalence. Besides, only very peculiar terms u_0 and u_1 are related by the general equivalence but not by Definition 2.4. These terms are like black holes: they reduce (in some context $\mathbb{C}$) to values $[a]v_0$ and $[b]v_1$ with $a \neq b$ that never evaluate their arguments. Indeed, if $\mathbb{E} = [c]\Box\ t_0 \dots t_n$, then $\mathbb{E}[\mu a.[a]v_0] \to_{\mathrm{n}} \mathbb{E}[v_0\langle^{\mathbb{E}}/_a\rangle]$, and $\mathbb{E}[\mu a.[b]v_1] \Downarrow_{\mathrm{n}} [b]v_1\langle^{\mathbb{E}}/_a\rangle$. Suppose that when evaluating $\mathbb{E}[v_0\langle^{\mathbb{E}}/_a\rangle]$, we evaluate one of the t_i's. Then by replacing t_i with Ω, we obtain a context $\mathbb{E}'$ such that $\mathbb{E}'[\mu a.[a]v_0] \Uparrow_{\mathrm{n}}$ (because Ω will be evaluated), and $\mathbb{E}'[\mu a.[b]v_1] \Downarrow_{\mathrm{n}}$, which is in contradiction with the fact that u_0 and u_1 are in the general equivalence (they are distinguished by $\mathbb{E}'[\mu a.\mathbb{C}]$).

We extend Definition 2.4 to any closed terms t_0, t_1, by saying that $t_0 \approx_c t_1$ if $[a]t_0 \approx_c [a]t_1$ for any fresh a. Other versions of the extension are possible, for example by replacing "for any a" by "for some a", or by dropping the freshness requirement; as can be shown using the results of Section 2.3, all these definitions are equivalent. We can also define contextual equivalence on open terms, using the notion of *open extension*, which extends any relation on closed (named) terms to open (named) terms. We say a substitution σ closes t (or u) if σ replaces the variables in $\mathsf{fv}(t)$ (or $\mathsf{fv}(u)$) with closed terms.

Definition 2.6 Let $\mathcal{R}$ be a relation on closed (named) terms. Two terms t_0 and t_1 are in the open extension of $\mathcal{R}$, written $t_0\ \mathcal{R}^\circ\ t_1$, if for all substitutions σ closing t_0 and t_1, we have $t_0\sigma\ \mathcal{R}\ t_1\sigma$ (and similarly for $u_0\ \mathcal{R}^\circ\ u_1$).

2.3 Applicative Bisimilarity

We propose a notion of applicative bisimulation, which tests values by applying them to a random closed argument. As with contextual equivalence, we give the definitions for named terms, before extending it to regular terms.

Definition 2.7 A relation $\mathcal{R}$ on closed named terms is an applicative bisimulation if $u_0\ \mathcal{R}\ u_1$ implies

- if $u_0 \to_{\mathrm{n}} u_0'$, then there exists u_1' such that $u_1 \to_{\mathrm{n}}^* u_1'$ and $u_0'\ \mathcal{R}\ u_1'$;
- if $u_0 = [a]\lambda x.t_0$, then there exists t_1 such that $u_1 \to_{\mathrm{n}}^* [a]\lambda x.t_1$ and for all t, we have $[a]t_0\langle^{[a]\Box\ t}/_a\rangle\{t/x\}\ \mathcal{R}\ [a]t_1\langle^{[a]\Box\ t}/_a\rangle\{t/x\}$;

- the symmetric conditions on u_1.

Applicative bisimilarity, written $\approx$, is the largest applicative bisimulation.

For regular terms, we write $t_0 \mathrel{\mathcal{R}} t_1$ if $[a]t_0 \mathrel{\mathcal{R}} [a]t_1$ for any $a \notin \mathsf{fn}(t_0, t_1)$. The first item of Definition 2.7 plays the bisimulation game for named terms which are not named values. If u_0 is a named value $[a]\lambda x.t_0$, then u_1 has to reduce to a named value $[a]\lambda x.t_1$, and we compare the values by applying them to an argument t. However, a context cannot interact with $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$ by simply applying them to t, because $([a]\lambda x.t_0)\ t$ is not allowed by the syntax. Consequently, we have to prefix them first with μa. As a result, we consider the named terms $[a](\mu a.[a]\lambda x.t_0)\ t$ and $[a](\mu a.[a]\lambda x.t_1)\ t$, which reduce to, respectively, $[a](\lambda x.t_0\langle {}^{[a]\Box\, t}/_{a}\rangle)t$ and $[a](\lambda x.t_1\langle {}^{[a]\Box\, t}/_{a}\rangle)t$, and then to $[a]t_0\langle {}^{[a]\Box\, t}/_{a}\rangle\{t/x\}$ and $[a]t_1\langle {}^{[a]\Box\, t}/_{a}\rangle\{t/x\}$; we obtain the terms in the clause for values of Definition 2.7.

Remark 2.8 When considering $[a](\mu a.[a]\lambda x.t_0)\ t$ and $[a](\mu a.[a]\lambda x.t_1)\ t$, we use the same top-level name a as the one of the named values $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$. We could use a fresh name b instead; reusing the same name makes the bisimulation proofs easier (we do not have to introduce unnecessary fresh names).

We can also define a big-step version of the bisimulation, where we consider only evaluation to a value.

Definition 2.9 A relation $\mathcal{R}$ on closed named terms is a big-step applicative bisimulation if $u_0 \mathrel{\mathcal{R}} u_1$ implies

- if $u_0 \to_{\mathrm{n}}^* [a]\lambda x.t_0$, then there exists t_1 such that $u_1 \to_{\mathrm{n}}^* [a]\lambda x.t_1$ and for all t, we have $[a]t_0\langle {}^{[a]\Box\, t}/_{a}\rangle\{t/x\} \mathrel{\mathcal{R}} [a]t_1\langle {}^{[a]\Box\, t}/_{a}\rangle\{t/x\}$;
- the symmetric condition on u_1.

Lemma 2.10 *If $\mathcal{R}$ is a big-step applicative bisimulation, then $\mathcal{R} \subseteq \approx$.*

As a first property, we prove that reduction (and therefore, evaluation) is included in bisimilarity.

Lemma 2.11 *We have $\to_{\mathrm{n}}^* \subseteq \approx$.*

Proof. By showing that $\{(u, u') \mid u \to_{\mathrm{n}}^* u'\} \cup \{(u, u)\}$ is a big-step bisimulation. □

We give a basic example to show how applicative bisimulation can be used.

Example 2.12 For all closed v and $a, b \notin \mathsf{fn}(v)$ [3], we prove that $[a]v \approx [a]\lambda x.\mu b.[a]v$ by showing that $\{([a]v, [a]\lambda x.\mu b.[a]v) \mid b \notin \mathsf{fn}(v)\} \cup \approx$ is an applicative bisimulation. Indeed, if $v = \lambda x.t$, then for all t', we have $[a]t\{t'/x\} \approx [a]\mu b.[a]v\ t'$, because $[a]\mu b.[a]v\ t' \to_{\mathrm{n}}^* [a]t\{t'/x\}$ (and by Lemma 2.11).

2.4 Soundness and Completeness

We now prove that $\approx$ coincides with $\approx_c$. We first show that $\approx$ is a congruence using Howe's method [8,7], which is a classic proof method to show that an applicative bisimilarity is a congruence. As in [10], we need to slightly adapt the proof to the

[3] Note that the result still holds if $a \in \mathsf{fn}(v)$.

$\lambda\mu$-calculus. Here we only sketch the application of the method, all the details can be found in [3, Appendix A.1].

The principle of the method is to prove that a relation called the *Howe's closure* of $\approx$, which is a congruence by construction, is also a bisimulation. The definition of Howe's closure relies on an auxiliary relation, called the *compatible refinement* $\widetilde{\mathcal{R}}$ of a relation $\mathcal{R}$, and inductively defined by the following rules:

$$\frac{}{x\ \widetilde{\mathcal{R}}\ x} \qquad \frac{t_0\ \mathcal{R}\ t_1}{\lambda x.t_0\ \widetilde{\mathcal{R}}\ \lambda x.t_1} \qquad \frac{t_0\ \mathcal{R}\ t_1 \qquad t'_0\ \mathcal{R}\ t'_1}{t_0\ t'_0\ \widetilde{\mathcal{R}}\ t_1\ t'_1} \qquad \frac{u_0\ \mathcal{R}\ u_1}{\mu a.u_0\ \widetilde{\mathcal{R}}\ \mu a.u_1}$$

$$\frac{t_0\ \mathcal{R}\ t_1}{[a]t_0\ \widetilde{\mathcal{R}}\ [a]t_1} \qquad \frac{t_0\ \mathcal{R}\ t_1 \qquad \mathbb{E}_0\ \widetilde{\mathcal{R}}\ \mathbb{E}_1}{t_0\langle\mathbb{E}_0/a\rangle\ \widetilde{\mathcal{R}}\ t_1\langle\mathbb{E}_1/a\rangle} \qquad \frac{u_0\ \mathcal{R}\ u_1 \qquad \mathbb{E}_0\ \widetilde{\mathcal{R}}\ \mathbb{E}_1}{u_0\langle\mathbb{E}_0/a\rangle\ \widetilde{\mathcal{R}}\ u_1\langle\mathbb{E}_1/a\rangle}$$

$$\frac{}{\Box\ \widetilde{\mathcal{R}}\ \Box} \qquad \frac{E_0\ \widetilde{\mathcal{R}}\ E_1 \qquad t_0\ \mathcal{R}\ t_1}{E_0\ t_0\ \widetilde{\mathcal{R}}\ E_1\ t_1} \qquad \frac{E_0\ \widetilde{\mathcal{R}}\ E_1}{[a]E_0\ \widetilde{\mathcal{R}}\ [a]E_1}$$

In the original definition of compatible refinement [7], two terms are related by $\widetilde{\mathcal{R}}$ if they have the same outer language constructor, and their subterms are related by $\mathcal{R}$. In the $\lambda\mu$-calculus, compatible refinement is extended to (named) evaluation contexts, and we allow for the substitution of names with related named contexts.

Given two relations $\mathcal{R}_1$ and $\mathcal{R}_2$, we write $\mathcal{R}_1\mathcal{R}_2$ for their composition, e.g., $t_0\ \mathcal{R}_1\mathcal{R}_2\ t_2$ holds if there exists t_1 such that $t_0\ \mathcal{R}_1\ t_1$ and $t_1\ \mathcal{R}_2\ t_2$. We can now define Howe's closure of $\approx$, written $\approx^\bullet$, as follows.

Definition 2.13 The Howe's closure $\approx^\bullet$ is the smallest relation verifying:

$$\approx^\circ\ \subseteq\ \approx^\bullet \qquad\qquad \approx^\bullet\approx^\circ\ \subseteq\ \approx^\bullet \qquad\qquad \widetilde{\approx^\bullet}\ \subseteq\ \approx^\bullet$$

Howe's closure is defined on open (named) terms as well as on (named) evaluation contexts. Because it contains its compatible refinement, $\approx^\bullet$ is a congruence. To prove it is a bisimulation, we need a stronger result, called a pseudo-simulation lemma, where we test named values not with the same argument, but with arguments t'_0, t'_1 related by $\approx^\bullet$.

Lemma 2.14 *Let $(\approx^\bullet)^c$ be $\approx^\bullet$ restricted to closed terms, and let $u_0\ (\approx^\bullet)^c\ u_1$.*

- *If $u_0 \to_{\mathrm{n}} u'_0$, then $u_1 \to_{\mathrm{n}}^* u'_1$ and $u'_0\ (\approx^\bullet)^c\ u'_1$.*
- *If $u_0 = [a]\lambda x.t_0$, then $u_1 \to_{\mathrm{n}}^* [a]\lambda x.t_1$ and for all $t'_0\ (\approx^\bullet)^c\ t'_1$, we have $[a]t_0\langle{}^{[a]\Box\, t'_0}/_a\rangle\{t'_0/x\}\ (\approx^\bullet)^c\ [a]t_1\langle{}^{[a]\Box\, t'_1}/_a\rangle\{t'_1/x\}$.*

With this result, we can prove that $(\approx^\bullet)^c$ is a bisimulation, and therefore included in $\approx$. Because it also contains $\approx$ by definition, we have $\approx = (\approx^\bullet)^c$, and this implies that $\approx$ is a congruence. As a result, $\approx$ is sound w.r.t. to $\approx_c$.

Theorem 2.15 $\approx\ \subseteq\ \approx_c$.

To simplify the proof of completeness (the reverse inclusion), we consider an alternate definition of contextual equivalence, where we test terms with named

evaluation contexts only. By doing so, we prove a context lemma in the process. [4]

Definition 2.16 Let u_0, u_1 be closed terms. We write $u_0 \mathrel{\dot{\approx}_c} u_1$ if for all closed contexts $\mathbb{E}$ and names a, there exist b, v_0, v_1 such that $\mathbb{E}[\mu a.u_0] \Downarrow_{\mathrm{n}} [b]v_0$ iff $\mathbb{E}[\mu a.u_1] \Downarrow_{\mathrm{n}} [b]v_1$.

Theorem 2.17 $\approx_c \subseteq \mathrel{\dot{\approx}_c} \subseteq \approx$.

The first inclusion is by definition, and the second one is by showing that $\mathrel{\dot{\approx}_c}$ is a big-step applicative bisimulation.

2.5 Comparison with Lassen's Work

In [10], Lassen also proposes a definition of applicative bisimilarity that he proves sound, but he conjectures that it is not complete. We discuss here the differences between the two approaches.

Lassen defines a notion of bisimulation for regular terms only, and not for named terms. The definition is as follows.

Definition 2.18 A relation $\mathcal{R}$ on closed terms is a Lassen applicative bisimulation if $t_0 \mathrel{\mathcal{R}} t_1$ implies:

- for all a, if $[a]t_0 \to_{\mathrm{n}}^* [b]\lambda x.t_0'$, then there exists t_1' such that $[a]t_1 \to_{\mathrm{n}}^* [b]\lambda x.t_1'$, and for all t, we have $t_0'\langle {}^{[b]\Box\, t}/_{b}\rangle\{t/x\} \mathrel{\mathcal{R}} t_1'\langle {}^{[b]\Box\, t}/_{b}\rangle\{t/x\}$;
- the symmetric condition on t_1.

Lassen's definition is quite similar to our definition of big-step applicative bisimulation (Definition 2.9), except it requires $t_0\langle {}^{[b]\Box\, t}/_{b}\rangle\{t/x\} \mathrel{\mathcal{R}} t_1\langle {}^{[b]\Box\, t}/_{b}\rangle\{t/x\}$, which implies that these terms must be related when reduced with any top-level name a. This is more restrictive than our definition, where we compare these terms only with the top-level name b (or, as discussed in Remark 2.8, we could instead compare $[c]t_0\langle {}^{[c]\Box\, t}/_{b}\rangle\{t/x\}$ and $[c]t_1\langle {}^{[c]\Box\, t}/_{b}\rangle\{t/x\}$ for some fresh name c). To illustrate the difference, we consider Lassen's counter-example from [10].

Example 2.19 Let $t_0 \stackrel{\text{def}}{=} (\lambda x.\lambda y.x\, x)\,(\lambda x.\lambda y.x\, x)$, and $t_1 \stackrel{\text{def}}{=} \mu a.[a]\lambda y.\mu c.[a]t_0$ (with $c \neq a$). These terms are not bisimilar according to Lassen's definition. For all b, we have $[b]t_0 \to_{\mathrm{n}}^* [b]\lambda y.t_0$ and $[b]t_1 \to_{\mathrm{n}}^* [b]\lambda y.\mu c.[b]t_0$. With Lassen's definition, one has to relate t_0 and $\mu c.[b]t_0\, t$ for any t, which means comparing $[d]t_0$ and $[d]\mu c.[b]t_0\, t$ for all d. But these two terms are not equivalent if $d \neq b$.

Lassen conjectures in [10] that these terms are contextually equivalent, and we can indeed prove that they are (big-step) bisimilar with our definition: we just have to compare $[b]t_0$ and $[b]\mu a'.[b]t_0\, t$ (or $[c]t_0$ and $[c]\mu a'.[c]t_0\, t$ for some fresh c) for any t, and both terms evaluate to $[b]\lambda x.t_0$ (or $[c]\lambda x.t_0$) and are therefore equivalent.

By comparing primarily named terms, as we do in our definition, we can keep track of what happens to the top level, and especially of any connection between the top level and a subterm. In Example 2.19, we can see that it is essential to remember that b represents the top level in $\mu c.[b]t_0\, t$, and therefore it does not make sense to

[4] We cannot directly use David and Py's context lemma [4], because we use a different notion of contextual equivalence.

compare $[d]t_0$ and $[d]\mu c.[b]t_0 t$ for any $d \neq b$, as we have to do with Lassen's definition. We believe that comparing named terms is essential to obtain completeness w.r.t. contextual equivalence; note that the sound and complete normal form bisimilarity for the $\lambda\mu\rho$-calculus [19] is also defined on named terms.

2.6 David and Py's Counter-Example

In [4], David and Py give a counter-example showing that Böhm's theorem fails in CBN $\lambda\mu$-calculus. They prove that their terms are contextually equivalent using a context lemma. Here we slightly simplify their counter-example, and prove equivalence using applicative bisimilarity. Note that these terms cannot be proved equivalent with (a CBN variant of) eager normal form bisimilarity [10,19].

Example 2.20 Let $0 \stackrel{\text{def}}{=} \lambda x.\lambda y.y$, $1 \stackrel{\text{def}}{=} \lambda x.\lambda y.x$, and $t_a \stackrel{\text{def}}{=} \mu c.[a]0$. Then we have $\lambda x.\mu a.[a]x\ \mu b.[a]x\ t_a\ 0 \approx \lambda x.\mu a.[a]x\ \mu b.[a]x\ t_a\ 1$ [5].

Proof. [Sketch] We only give the main ideas here, the complete equivalence proof can be found in [3, Appendix A.2]. First, $\lambda x.\mu a.[a]x\ \mu b.[a]x\ t_a\ 0$ is not normal form bisimilar to $\lambda x.\mu a.[a]x\ \mu b.[a]x\ t_a\ 1$, because the subterms of these two terms are not normal form bisimilar (0 is not equivalent to 1).

To prove applicative bisimilarity, let c be a fresh name and t be a closed term. We want to relate $[c]\mu a.[a]t\,\mu b.[a]t\,t_a 0$ and $[c]\mu a.[a]t\,\mu b.[a]t\,t_a 1$, which reduce respectively to $[c]t\,\mu b.[c]t\,t_c 0$ (1) and $[c]t\,\mu b.[c]t\,t_c 1$ (2). Let $d \notin \mathsf{fn}(t)$; we distinguish several cases depending on the behavior of $[d]t$. The interesting case is when $[d]t \Downarrow_{\mathrm{n}} [d]\lambda y.t'$; then $\mu b.[c]t\ t_c\ 0$ or $\mu b.[c]t\ t_c\ 1$ is passed as an argument to $\lambda y.t'$ in respectively (1) and (2). If t' executes its argument (that is, if t' reduces to $E[y]$ for some E), then (1) reduces to $[c]t\,t_c\,0$ (3), and (2) to $[c]t\,t_c\,1$ (4). But we know that $[d]t \Downarrow_{\mathrm{n}} [d]\lambda y.t'$, and t' executes its argument, so when evaluating (3) and (4), t_c will be reduced, and therefore (3) and (4) will evaluate to $[c]0$.

In the other cases (e.g., $[d]t \Downarrow_{\mathrm{n}} [e]\lambda y.t'$ with $e \neq d$), either (1) and (2) eventually get to a point similar to the situation above where t_c is executed, or they diverge. In all cases, they are applicative bisimilar. □

3 Call-by-Value $\lambda\mu$-calculus

3.1 Semantics and Contextual Equivalence

In this section, we use CBV left-to-right evaluation, which is encoded in the syntax of the CBV evaluation contexts:

$$E ::= \Box \mid E\,t \mid v\,E$$

[5] The terms David and Py consider in their work are $\lambda x.\mu a.[a]x\mu b.[a](x t_a 0)t_a$ and $\lambda x.\mu a.[a]x\mu b.[a](x t_a 1)t_a$. However, the additional argument t_a would not come into play in the proof we present, so we have elided it.

The CBV reduction relation $\to_{\mathrm{v}}$ is defined by the following rules.

$$
\begin{array}{rrcll}
(\beta_v) & [a](\lambda x.t)\, v & \to_{\mathrm{v}} & [a]t\{v/x\} & \\
(\mu) & [a]\mu b.u & \to_{\mathrm{v}} & u\langle [a]\Box/b\rangle & \\
(app) & [a]t_0\, t_1 & \to_{\mathrm{v}} & u\langle [a]\Box\, t_1/b\rangle & \text{if } [b]t_0 \to_{\mathrm{v}} u \text{ and } b \notin \mathsf{fn}([a]t_0\, t_1) \\
(app_v) & [a]v\, t & \to_{\mathrm{v}} & u\langle [a]v\, \Box/b\rangle & \text{if } [b]t \to_{\mathrm{v}} u \text{ and } b \notin \mathsf{fn}([a]v\, t)
\end{array}
$$

With rule (app_v), we reduce arguments to values, to be able to apply CBV β-reduction (rule (β_v)). The rules (μ) and (app) are unchanged. We could also express reduction with top-level named evaluation contexts, as in Lemma 2.1. Furthermore, CBV reduction is compatible with CBV contexts, as in Lemma 2.2. We write $\to_{\mathrm{v}}^*$ for the reflexive and transitive closure of $\to_{\mathrm{v}}$, $\Downarrow_{\mathrm{v}}$ for CBV evaluation, and $\Uparrow_{\mathrm{v}}$ for CBV divergence.

We use the same definition of contextual equivalence as in CBN.

Definition 3.1 Let u_0, u_1 be closed named terms. We write $u_0 \approx_c u_1$, if for all closed contexts $\mathbb{C}$ and names a, there exist b, v_0, and v_1 such that $\mathbb{C}[\mu a.u_0] \Downarrow_{\mathrm{v}} [b]v_0$ iff $\mathbb{C}[\mu a.u_1] \Downarrow_{\mathrm{v}} [b]v_1$.

However, unlike in CBN, this definition (where we require the resulting values to have the same top-level names) coincides with the general definition where we simply say "$\mathbb{C}[\mu a.u_0] \Downarrow_{\mathrm{v}}$ iff $\mathbb{C}[\mu a.u_1] \Downarrow_{\mathrm{v}}$." Indeed, if $\mathbb{C}[\mu a.u_0] \Downarrow_{\mathrm{v}} [b]v_0$ and $\mathbb{C}[\mu a.u_1] \Downarrow_{\mathrm{v}} [c]v_1$ with $c \neq b$, then we can easily distinguish them, because $[b]\mu b.\mathbb{C}[\mu a.u_0]\ \Omega \to_{\mathrm{v}}^* [b]v_0\langle [b]\Box\,\Omega/b\rangle\ \Omega \Uparrow_{\mathrm{v}}$, and $[b]\mu b.\mathbb{C}[\mu a.u_1]\ \Omega \Downarrow_{\mathrm{v}} [c]v_1\langle [b]\Box\,\Omega/b\rangle$.

We extend $\approx_c$ to any closed terms as in CBN. The definition of open extension is slightly changed in CBV, compared to CBN: we close open terms by substituting their variables with closed *values* only, and not any closed terms.

3.2 Applicative Bisimilarity

Before giving its complete definition, we explain how applicative bisimilarity $\approx$ should compare two named values $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$. The following reasoning explains and justifies the clauses in Definition 3.4. In particular, we provide counter-examples to show that we cannot simplify this definition.

In CBV λ-calculus (and also with delimited control [2]), values are tested by applying them to an arbitrary value argument. Following this principle, it is natural to propose the following clause for CBV $\lambda\mu$-calculus.

(1) For all v, we have $[a]t_0\langle [a]\Box\, v/a\rangle\{v/x\} \approx [a]t_1\langle [a]\Box\, v/a\rangle\{v/x\}$.

As with Definition 2.7, we in fact compare $[a](\mu a.[a]\lambda x.t_0)\, v$ with $[a](\mu a.[a]\lambda x.t_1)\, v$, which reduce to the terms in clause (1). However, such a clause would produce an unsound applicative bisimilarity; it would relate terms that are not contextually equivalent, like the ones in the next example.

Example 3.2 Let $v_0 \stackrel{\text{def}}{=} \lambda x.\mu b.[a]w\ x$, $v_1 \stackrel{\text{def}}{=} \lambda x.w\ x\ x$, with $w \stackrel{\text{def}}{=} \lambda y.\lambda z.z\ y$. Then we have $([a]v_0)\langle [a]\Box\, v/a\rangle = [a](\lambda x.\mu b.[a]w\ x\ v)\ v \to_{\mathrm{v}}^* [a]w\ v\ v$ and $([a]v_1)\langle [a]\Box\, v/a\rangle = [a](\lambda x.w\ x\ x)\ v \to_{\mathrm{v}}^* [a]w\ v\ v$. Because they reduce to the same term, $([a]v_0)\langle [a]\Box\, v/a\rangle$

is contextually equivalent to $([a]v_1)\langle {}^{[a]\Box\, v}/_{a}\rangle$, and using clause (1) would lead us to conclude that $[a]v_0$ and $[a]v_1$ are equivalent as well.

However, $[a]v_0$ and $[a]v_1$ can be distinguished with $t \stackrel{\text{def}}{=} \mu d.[d]\lambda y.\mu c.[d]w'$, where $w' \stackrel{\text{def}}{=} \lambda x.\mu c.[d]x\ w''$ and $w'' \stackrel{\text{def}}{=} \lambda x.\lambda y.\lambda z.\Omega$. Indeed, we can check that $([a]v_0)\langle {}^{[a]\Box\, t}/_{a}\rangle \to_{\mathrm{v}}^{*} \lambda z.\Omega$ and $([a]v_1)\langle {}^{[a]\Box\, t}/_{a}\rangle \to_{\mathrm{v}}^{*} \Omega$. This discrepancy comes from the fact that, in $([a]v_1)\langle {}^{[a]\Box\, t}/_{a}\rangle$, t is reduced to a value once, capturing $[a]v_1\,\Box$ in the process, while t is reduced twice to a value in $([a]v_0)\langle {}^{[a]\Box\, t}/_{a}\rangle$, and each time it captures a different context. Therefore, $[a]v_0$ and $[a]v_1$ are distinguished by the context $[a](\mu a.\Box)\,t$, and they are consequently not contextually equivalent.

Example 3.2 suggests that we should compare $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$ with contexts of the form $[a]\Box\, t$, instead of $[a]\Box\, v$. Therefore, we should compare $u_0 \stackrel{\text{def}}{=} [a](\lambda x.t_0\langle {}^{[a]\Box\, t}/_{a}\rangle)\ t$ with $u_1 \stackrel{\text{def}}{=} [a](\lambda x.t_1\langle {}^{[a]\Box\, t}/_{a}\rangle)\ t$. However, we can restrict a bit the choice of the testing term t, based on its behavior. Let $b \notin \mathsf{fn}(t)$; if $[b]t$ diverges, then u_0 and u_1 diverge as well, and we gain no information on $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$ themselves. If $[b]t \to_{\mathrm{v}}^{*} [c]v$ with $b \neq c$, then $u_0 \to_{\mathrm{v}}^{*} [c]v\langle {}^{[a]\lambda x.t_0\langle {}^{[a]\Box\, t}/_{a}\rangle\,\Box}/_{b}\rangle$, and similarly with u_1. The values $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$ are captured by $[b]t$, and no interaction between t and the two named values takes place in the process ($[a]\lambda x.t_0$ and $[a]\lambda x.t_1$ are not applied to any value); again, we do not gain any new knowledge on the behavior of $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$. Finally, if $[b]t \to_{\mathrm{v}}^{*} [b]v$, then $u_0 \to_{\mathrm{v}}^{*} [b](\lambda x.t_0\langle {}^{[a]\Box\, t}/_{a}\rangle)\ v\langle {}^{[a]\lambda x.t_0\langle {}^{[a]\Box\, t}/_{a}\rangle\,\Box}/_{b}\rangle$, and similarly with u_1; in this case, a value is indeed passed to $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$, and we can compare their respective behaviors. Therefore, an interaction happens between t and the tested values iff $[b]t \to_{\mathrm{v}}^{*} [b]v$, and the results of the interaction (after β-reduction) are the two terms in the clause below.

(2) For all t, b, v such that $[b]t \to_{\mathrm{v}}^{*} [b]v$ and $b \notin \mathsf{fn}(t)$, we have

$$[a]t_0\langle {}^{[a]\Box\, t}/_{a}\rangle\{v\langle {}^{[a]\lambda x.t_0\langle {}^{[a]\Box\, t}/_{a}\rangle\,\Box}/_{b}\rangle/x\} \approx [a]t_1\langle {}^{[a]\Box\, t}/_{a}\rangle\{v\langle {}^{[a]\lambda x.t_1\langle {}^{[a]\Box\, t}/_{a}\rangle\,\Box}/_{b}\rangle/x\}.$$

Unfortunately, clause (2) is not enough to obtain a sound bisimilarity. The next example shows that an extra clause is needed.

Example 3.3 Let $v_0 \stackrel{\text{def}}{=} \lambda x.\mu b.[a](\lambda y.\lambda z.wy)x$ and $v_1 \stackrel{\text{def}}{=} w$ with $w \stackrel{\text{def}}{=} \lambda x.w'(x\lambda y.y)$, and $w' \stackrel{\text{def}}{=} \lambda y.y\ \lambda z.\Omega$. We first show that $[a]v_0$ and $[a]v_1$ are related by clause (2). Let t such that $[b]t \Downarrow_{\mathrm{v}} [b]v$ for $b \notin \mathsf{fn}(t)$. Then we have $([a]v_0)\langle {}^{[a]\Box\, t}/_{a}\rangle \to_{\mathrm{v}}^{*} [a]w'\ (v\langle {}^{[a]v_0\langle {}^{[a]\Box\, t}/_{a}\rangle\,\Box}/_{b}\rangle\ \lambda x.x)$ and $([a]v_1)\langle {}^{[a]\Box\, t}/_{a}\rangle \to_{\mathrm{v}}^{*} [a]w'\ (v\langle {}^{[a]v_1\,\Box}/_{b}\rangle\ \lambda x.x)$. We can prove that the two resulting terms are contextually equivalent by showing that the relation $\{(u\langle {}^{[a]E[v_0\langle {}^{[a]E[\Box\, t]}/_{a}\rangle\,\Box]}/_{b}\rangle, u\langle {}^{[a]E[v_1\,\Box]}/_{b}\rangle) \mid [b]t \Downarrow_{\mathrm{v}} [b]v, b \notin \mathsf{fn}(t)\}$ is an applicative bisimulation according to Definition 3.4, and by using Theorem 3.9 (see [3, Appendix B.1]). Because $([a]v_0)\langle {}^{[a]\Box\, t}/_{a}\rangle$ and $([a]v_1)\langle {}^{[a]\Box\, t}/_{a}\rangle$ are contextually equivalent, using only clause (2) would lead us to conclude that $[a]v_0$ and $[a]v_1$ are also equivalent.

However, these two named values can be distinguished with the context $[a](\lambda x.x\ x)\,\mu a.\Box$, because in one case we have $([a]v_0)\langle {}^{[a](\lambda x.x\ x)\,\Box}/_{a}\rangle \to_{\mathrm{v}}^{*} \lambda z.\Omega$, and in the other $([a]v_1)\langle {}^{[a](\lambda x.x\ x)\,\Box}/_{a}\rangle \to_{\mathrm{v}}^{*} \Omega$. As in Example 3.2, when evaluating $([a]v_0)\langle {}^{[a](\lambda x.x\ x)\,\Box}/_{a}\rangle$, the body of v_0 is evaluated twice, and two different contexts

are captured each time. In contrast, v_1 does not contain any control effect, so when its body is evaluated twice, we get the same result.

Example 3.3 shows that we have to compare two values $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$ by also testing them with contexts of the form $[a]v\,\Box$, i.e., by considering $[a]v\ \lambda x.t_0\langle{}^{[a]v\,\Box}/_{a}\rangle$ and $[a]v\ \lambda x.t_1\langle{}^{[a]v\,\Box}/_{a}\rangle$. If $v = \lambda x.t$, then these terms reduce in one β-reduction step into $[a]t\{\lambda x.t_0\langle{}^{[a]v\,\Box}/_{a}\rangle/x\}$, and $[a]t\{\lambda x.t_1\langle{}^{[a]v\,\Box}/_{a}\rangle/x\}$. Taking this and clause (2) into account, we obtain the following definition of applicative bisimulation.

Definition 3.4 A relation $\mathcal{R}$ on closed named terms is an applicative bisimulation if $u_0\ \mathcal{R}\ u_1$ implies

- if $u_0 \to_{\mathrm{v}} u_0'$, then there exists u_1' such that $u_1 \to_{\mathrm{v}}^* u_1'$ and $u_0'\ \mathcal{R}\ u_1'$;
- if $u_0 = [a]\lambda x.t_0$, then there exists t_1 such that $u_1 \to_{\mathrm{v}}^* [a]\lambda x.t_1$, and:

(i) for all t, b, v such that $[b]t \to_{\mathrm{v}}^* [b]v$ and $b \notin \mathsf{fn}(t)$, we have

$$[a]t_0\langle{}^{[a]\Box\,t}/_{a}\rangle\{v\langle{}^{[a]\lambda x.t_0\langle{}^{[a]\Box\,t}/_{a}\rangle\,\Box}/_{b}\rangle/x\}\ \mathcal{R}\ [a]t_1\langle{}^{[a]\Box\,t}/_{a}\rangle\{v\langle{}^{[a]\lambda x.t_1\langle{}^{[a]\Box\,t}/_{a}\rangle\,\Box}/_{b}\rangle/x\};$$

(ii) for all $v = \lambda x.t$, we have

$$[a]t\{\lambda x.t_0\langle{}^{[a]v\,\Box}/_{a}\rangle/x\}\ \mathcal{R}\ [a]t\{\lambda x.t_1\langle{}^{[a]v\,\Box}/_{a}\rangle/x\};$$

- the symmetric conditions on u_1.

Applicative bisimilarity, written $\approx$, is the largest applicative bisimulation.

The definition is extended to regular terms t_0, t_1 as in CBN, by using a fresh top-level name a. Note that clause (ii) implies that a bisimulation $\mathcal{R}$ is a congruence w.r.t. (regular) values; indeed, if $v_0\ \mathcal{R}\ v_1$, then $[a]v_0\ \mathcal{R}\ [a]v_1$ for a fresh a, and so we have $[a]t\{v_0/x\}\ \mathcal{R}\ [a]t\{v_1/x\}$ for all t (by clause (ii)). This property simplifies the congruence proof of $\approx$ with Howe's method.

As in CBN, we can define a big-step version of the bisimulation (where we use evaluation instead of reduction), and bisimilarity contains reduction.

Lemma 3.5 *We have* $\to_{\mathrm{v}}^* \subseteq \approx$.

The applicative bisimulation for CBV is more difficult to use than the one for CBN, as we can see by considering again the terms of Example 2.12.

Example 3.6 Let $v = \lambda x.t$ and $a, b \notin \mathsf{fn}(v)$; then $[a]v \approx [a]\lambda x.\mu b.[a]v$. To prove clause (i), we consider t' be such that $[b]t' \to_{\mathrm{v}}^* [b]v'$ for $b \notin \mathsf{fn}(t')$; we have to compare $[a]t\{v'\langle{}^{[a]v\,\Box}/_{b}\rangle/x\}$ with $[a]\mu b.[a]v\ t'$. But $[a]\mu b.[a]v\ t' \to_{\mathrm{v}} [a]v\ t' \to_{\mathrm{v}}^* [a]t\{v'\langle{}^{[a]v\,\Box}/_{b}\rangle/x\}$, therefore we can conclude with Lemma 3.5.

For clause (ii), we have to relate $[a]t'\{v/y\}$ and $[a]t'\{\lambda x.\mu b.[a]v'\ v/y\}$ for all $v' = \lambda y.t'$. We proceed by case analysis on t'; the most interesting case is $t' = E[yv'']$. In this case, we have $[a]t'\{\lambda x.\mu b.[a]v'\ v/y\} \to_{\mathrm{v}}^* [a]v'\ v \to_{\mathrm{v}} [a]t'\{v/y\}$, therefore we can conclude with Lemma 3.5. To handle all the possible cases, we prove in [3, Appendix B.1] that $\{(u\{v/y\}, u\{\lambda x.\mu b.[a]t_0/y\}) \mid [a]t_0 \to_{\mathrm{v}}^* u\{v/y\}\} \cup \approx$ is an applicative bisimulation.

In the next example, we give two terms that can be proved equivalent with applicative bisimilarity but not with eager normal form bisimilarity [19].

Example 3.7 Let $u_0 \stackrel{\text{def}}{=} [b]\lambda xy.\Omega$, $v \stackrel{\text{def}}{=} \lambda y.\mu a.[b]\lambda x.y$, and $u_1 \stackrel{\text{def}}{=} [b]\lambda xy.\Theta_{\mathsf{v}}\ v\ y$, where $\Theta_{\mathsf{v}} \stackrel{\text{def}}{=} (\lambda xy.y\ (\lambda z.xx\ y\ z))\ (\lambda xy.y\ (\lambda z.xx\ y\ z))$ is Turing's call-by-value fixed-point combinator. For u_0 and u_1 to be normal form bisimilar, we need $[c]\Omega$ to be related to $[c]\Theta_{\mathsf{v}} v\, y$ for a fresh c, but $[c]\Theta_{\mathsf{v}} v\, y \Downarrow_{\mathrm{v}} [b]\lambda y.\Theta_{\mathsf{v}} v\, y$ and $[c]\Omega \Uparrow_{\mathrm{v}}$. In contrast, we can prove that $u_0 \approx u_1$ (see [3, Appendix B.1]).

We now briefly sketch the proofs of soundness and completeness; more details can be found in [3, Appendix B.2]. The application of Howe's method is easier than in CBN because, as already pointed out, an applicative bisimulation (and, therefore, the applicative bisimilarity) is already a congruence for regular values by definition. What is left to prove is congruence for (named) terms. We use the same definitions of compatible refinement and Howe's closure $\approx^\bullet$ as in CBN. However, because $\approx$ is a congruence for values, we can prove directly that the restriction of $\approx^\bullet$ to closed terms (written $(\approx^\bullet)^c$) is an applicative bisimulation, without having to prove a pseudo-simulation lemma (similar to Lemma 2.14) beforehand.

Lemma 3.8 *The relation $(\approx^\bullet)^c$ is an applicative bisimulation.*

As in CBN, we can conclude that $(\approx^\bullet)^c{=}\approx$, and therefore $\approx$ is a congruence. We can then deduce that $\approx$ is sound w.r.t. $\approx_c$. For the reverse inclusion, we use an alternate definition of contextual equivalence where we test terms with evaluation contexts (see Definition 2.16), and we prove it is an applicative bisimulation. As a result, $\approx$ coincides with $\approx_c$.

Theorem 3.9 $\approx{=}\approx_c$.

Remark 3.10 In [9], Koutavas *et al.* show that applicative bisimilarity cannot be sound in a CBV λ-calculus with exceptions, a mechanism that can be seen as a form of control. Our work agrees with their conclusions, as their definition of applicative bisimilarity compares λ-abstractions by applying them to values only, and Example 3.2 shows that it is indeed not sufficient.

3.3 Examples

Even if applicative bisimulation for CBV is difficult to use, we can still prove some equivalences with it. Here we give some examples inspired from Sabry and Felleisen's axiomatization of call/cc [16]. Given a name a, we write $a^\dagger$ for the term $\lambda x.\mu b.[a]x$, and we encode call/cc into $\lambda x.\mu a.[a]x\ a^\dagger$. Given a named context $\mathbb{E}$, we also write $\mathbb{E}^\dagger$ for $\lambda x.\mu b.\mathbb{E}[x]$, where $b \notin \mathsf{fn}(\mathbb{E})$. The first example is the axiom C_{tail} of [16], where call/cc is exchanged with a λ-abstraction.

Example 3.11 If $y \notin \mathsf{fv}(t_1)$ and b is fresh, then $[b](\lambda x.\mu a.[a]x\ a^\dagger)\ (\lambda y.(\lambda z.t_0)\ t_1) \approx [b](\lambda z.(\lambda x.\mu a.[a]x\ a^\dagger)\ (\lambda y.t_0))\ t_1$.

Proof. Let $v_0 \stackrel{\text{def}}{=} \lambda z.t_0\{b^\dagger/y\}$ and $v_1 \stackrel{\text{def}}{=} \lambda z.(\lambda x.\mu a.[a]x\, a^\dagger)\,(\lambda y.t_0)$. The term on the left reduces to $[b]v_0\ t_1$, so we relate this term to the one the right, i.e., $[b]v_1\ t_1$. We distinguish several cases depending on t_1. Let c be a fresh name. If $[c]t_1 \Downarrow_{\mathrm{v}} [c]v$,

then $[b]v_0\,t_1 \to_{\mathrm{v}}^* [b]t_0\{b^\dagger/y\}\{v\langle{}^{[b]v_0\,\Box}/_c\rangle/z\}$ and $[b]v_1\,t_1 \to_{\mathrm{v}}^* [b]t_0\{b^\dagger/y\}\{v\langle{}^{[b]v_1\,\Box}/_c\rangle/z\}$; because c is fresh, it does not occur in t_0, and the previous terms can be written $u\langle{}^{[b]v_0\,\Box}/_c\rangle$ and $u\langle{}^{[b]v_1\,\Box}/_c\rangle$ with $u \stackrel{\text{def}}{=} [b]t_0\{b^\dagger/y\}\{v/z\}$.

Similarly, if $[c]t_1 \Downarrow_{\mathrm{v}} [d]v$ with $c \neq d$, then $[b]v_0\,t_1 \Downarrow_{\mathrm{v}} [d]v\langle{}^{[b]v_0\,\Box}/_c\rangle$ and $[b]v_1\,t_1 \Downarrow_{\mathrm{v}} [d]v\langle{}^{[b]v_1\,\Box}/_c\rangle$. When testing these two values with clauses (i) and (ii), we obtain each time terms of the form $u\langle{}^{[b]v_0\,\Box}/_c\rangle$ and $u\langle{}^{[b]v_1\,\Box}/_c\rangle$ for some u. With this reasoning, we can prove that $\{(u\langle{}^{[b]v_0\,\Box}/_c\rangle, u\langle{}^{[b]v_1\,\Box}/_c\rangle) \mid u \in U^0\}$ is an applicative bisimulation, by case analysis on u. □

With the next example and congruence of $\approx$, we can prove the axiom C_{abort}.

Example 3.12 Let $a \neq b$; we have $[b]E[a^\dagger\,t] \approx [b]a^\dagger\,t$.

Proof. We prove that the relation $\mathcal{R} \stackrel{\text{def}}{=} \{(u\langle{}^{[b]E[\mathbb{E}^\dagger\,\Box]}/_c\rangle, u\langle{}^{[b]\mathbb{E}^\dagger\,\Box}/_c\rangle) \mid u \in U^0\}$ is an applicative bisimulation by case analysis on u. For example, if $u = [c]t$ and $u \Downarrow_{\mathrm{v}} [c]v$, then $u\langle{}^{[b]E[\mathbb{E}^\dagger\,\Box]}/_c\rangle \to_{\mathrm{v}}^* [b]E[\mathbb{E}^\dagger\,v\langle{}^{[b]E[\mathbb{E}^\dagger\,\Box]}/_c\rangle] \to_{\mathrm{v}} \mathbb{E}[v\langle{}^{[b]E[\mathbb{E}^\dagger\,\Box]}/_c\rangle]$ and $u\langle{}^{[b]\mathbb{E}^\dagger\,\Box}/_c\rangle \to_{\mathrm{v}}^* [b]\mathbb{E}^\dagger\,v\langle{}^{[b]\mathbb{E}^\dagger\,\Box}/_c\rangle \to_{\mathrm{v}} \mathbb{E}[v\langle{}^{[b]\mathbb{E}^\dagger\,\Box}/_c\rangle]$. If $\mathbb{E} \neq [d]\Box$ for all d, then the resulting terms are in $\mathcal{R}$, otherwise we get two named values; when checking clauses (i) and (ii), we obtain terms of the form $u'\langle{}^{[b]E[\mathbb{E}'^\dagger\,\Box]}/_c\rangle$ and $u'\langle{}^{[b]\mathbb{E}'^\dagger\,\Box}/_c\rangle$ that are in $\mathcal{R}$. The remaining cases are similar. □

Example 3.13 [axiom C_{lift}] We have $[b]E[(\lambda x.\mu a.[a]x\,a^\dagger)\,t] \approx [b]E[t\,(\lambda x.b^\dagger\,E[x])]$.

Proof. In this proof, we use an intermediary result, proved in [3, Appendix B.1]: if $\mathbb{E} = \mathbb{E}_0[E_1]$, then $\mathbb{E}^\dagger \approx \lambda x.\mathbb{E}_0^\dagger\,(E_1[x])$. The proof of the axiom itself is by case analysis on t. An interesting case is when $[d]t \Downarrow_{\mathrm{v}} [d]\lambda y.t'$ where $d \notin \mathsf{fn}(t)$. Then $[b]E[(\lambda x.\mu a.[a]x\,a^\dagger)\,t] \to_{\mathrm{v}}^* [b]E[(\lambda x.\mu a.[a]x\,a^\dagger)\,\lambda y.t'\langle{}^{[b]E[(\lambda x.\mu a.[a]x\,a^\dagger)\,\Box]}/_d\rangle] \to_{\mathrm{v}}^* [b]E[t'\langle{}^{[b]E[(\lambda x.\mu a.[a]x\,a^\dagger)\,\Box]}/_d\rangle\{\mathbb{E}^\dagger/y\}]$ (with $\mathbb{E} = [b]E$), and $[b]E[t\,(\lambda x.b^\dagger\,E[x])] \to_{\mathrm{v}}^* [b]E[t'\langle{}^{[b]E[\Box\,(\lambda x.b^\dagger\,E[x])]}/_d\rangle\{\lambda x.b^\dagger\,E[x]/y\}]$. From the intermediary result, and because $\approx$ is a congruence, we know that $[b]E[t'\{\mathbb{E}^\dagger/y\}] \approx [b]E[t'\{\lambda x.b^\dagger\,E[x]/y\}]$. Hence, to conclude the proof, one can show that

$$\{(u_0\langle{}^{\mathbb{E}[(\lambda x.\mu a.[a]x\,a^\dagger)\,\Box]}/_d\rangle, u_1\langle{}^{\mathbb{E}[\Box\,(\lambda x.\mathbb{E}_0^\dagger\,E_1[x])]}/_d\rangle) \;\mid\; u_0 \approx u_1, \mathbb{E} = \mathbb{E}_0[E_1]\}$$

is an applicative bisimulation. □

4 Conclusion

In this work we propose a definition of applicative bisimilarity for CBN and CBV $\lambda\mu$-calculus. Even if the two definitions seem quite different, they follow the same principles. First, we believe it is essential for completeness to hold to relate primarily named terms, and then extend the definition to all terms, as explained when discussing Lassen's definition of applicative bisimilarity (Section 2.5). The top-level names allow to keep track of how the top level is captured and manipulated in the compared terms.

Then, the idea is to test named values with elementary contexts, $[a]\Box\,t$ for CBN, and $[a]\Box\,t$ and $[a]v\,\Box$ for CBV. In the CBV case, we slightly restrict the terms t tested when considering $[a]\Box\,t$, but the resulting definition remains complex to use

compared to CBN, as we can see with Examples 2.12 and 3.6. However, we provide counter-examples showing that we cannot simplify it further (see Examples 3.2 and 3.3). In CBV as well as in CBN, applicative bisimilarity is harder to use than eager normal form bisimilarity [19], but our relations are complete characterizations of contextual equivalence, and we can therefore prove equivalences of terms that cannot be related with normal form bisimilarity, such as David and Py's example (see Example 2.20) and Example 3.7. To prove the equivalence between two given $\lambda\mu$-terms, one should start with the bisimulation of [19], and if it fails, try next our applicative (or environmental [3]) bisimulations.

We believe the relations we define remain complete w.r.t. contextual equivalence in other variants of the $\lambda\mu$-calculus (perhaps with some slight variations), such as $\lambda\mu$ with different reduction semantics (like, e.g., in [4]), typed $\lambda\mu$-calculus [15], or de Groote's extended calculus ($\Lambda\mu$-calculus [5]). However, any direct implications of this work for other calculi for abortive continuations such as the syntactic theory of control [6] are unclear and remain to be investigated. The reason is that our approach hinges on the syntactic notion of names, unique to the $\lambda\mu$-calculus, that allows one to keep track of the whereabouts of the top level.

Acknowledgments: We thank Wojciech Jedynak and the anonymous referees for helpful comments on the presentation of this work. The first author has been supported by the Polish NCN grant number DEC-011/03/B/ST6/00348.

References

[1] S. Abramsky. The lazy lambda calculus. In D. A. Turner, editor, *Research Topics in Functional Programming*, pages 65–116. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[2] D. Biernacki and S. Lenglet. Applicative bisimulations for delimited-control operators. In L. Birkedal, editor, *FOSSACS'12*, number 7213 in LNCS, pages 119–134, Tallinn, Estonia, Mar. 2012. Springer-Verlag.

[3] D. Biernacki and S. Lenglet. Sound and complete bisimilarities for call-by-name and call-by-value $\lambda\mu$-calculus. Research report RR-8447, Inria, Nancy, France, Jan. 2014. Available at `http://hal.inria.fr/hal-00926100`.

[4] R. David and W. Py. $\lambda\mu$-calculus and Böhm's theorem. *Journal of Symbolic Logic*, 66(1):407–413, 2001.

[5] P. de Groote. On the relation between the $\lambda\mu$-calculus and the syntactic theory of sequential control. In F. Pfenning, editor, *LPAR'94*, number 822 in LNAI, pages 31–43, Kiev, Ukraine, July 1994. Springer-Verlag.

[6] M. Felleisen and R. Hieb. The revised report on the syntactic theories of sequential control and state. *Theoretical Computer Science*, 103(2):235–271, 1992.

[7] A. D. Gordon. Bisimilarity as a theory of functional programming. *Theoretical Computer Science*, 228(1-2):5–47, 1999.

[8] D. J. Howe. Proving congruence of bisimulation in functional programming languages. *Information and Computation*, 124(2):103–112, 1996.

[9] V. Koutavas, P. B. Levy, and E. Sumii. From applicative to environmental bisimulation. *Electronic Notes in Theoretical Computer Science*, 276:215–235, 2011.

[10] S. B. Lassen. Bisimulation for pure untyped $\lambda\mu$-caluclus (extended abstract). Unpublished note, Jan. 1999.

[11] S. B. Lassen. Eager normal form bisimulation. In P. Panangaden, editor, *LICS'05*, pages 345–354, Chicago, IL, June 2005. IEEE Computer Society Press.

[12] S. B. Lassen. Head normal form bisimulation for pairs and the $\lambda\mu$-calculus. In R. Alur, editor, *LICS'06*, pages 297–306, Seattle, WA, Aug. 2006. IEEE Computer Society Press.

[13] J. H. Morris. *Lambda Calculus Models of Programming Languages*. PhD thesis, Massachusets Institute of Technology, 1968.

[14] C.-H. L. Ong and C. A. Stewart. A Curry-Howard foundation for functional computation with control. In N. D. Jones, editor, *POPL*, pages 215–227, Paris, France, Jan. 1997. ACM Press.

[15] M. Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In A. Voronkov, editor, *LPAR'92*, number 624 in LNAI, pages 190–201, St. Petersburg, Russia, July 1992. Springer-Verlag.

[16] A. Sabry and M. Felleisen. Reasoning about programs in continuation-passing style. *Lisp and Symbolic Computation*, 6(3/4):289–360, 1993.

[17] D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. In A. Scedrov, editor, *LICS'92*, pages 102–109, Santa Cruz, California, June 1992. IEEE Computer Society.

[18] D. Sangiorgi, N. Kobayashi, and E. Sumii. Environmental bisimulations for higher-order languages. *ACM Transactions on Programming Languages and Systems*, 33(1):1–69, Jan. 2011.

[19] K. Støvring and S. B. Lassen. A complete, co-inductive syntactic theory of sequential control and state. In M. Felleisen, editor, *POPL'07*, SIGPLAN Notices, Vol. 42, No. 1, pages 161–172, Nice, France, Jan. 2007. ACM Press.